(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 543 560 A2**

(12) **EUROPEAN PATENT APPLICATION**

(21) Application number: **92310300.6**

(51) Int. Cl.[5]: **G06F 15/16**

(22) Date of filing: **11.11.92**

(30) Priority: **19.11.91 US 794573**

(43) Date of publication of application:
26.05.93 Bulletin 93/21

(84) Designated Contracting States:
DE FR GB IT

(71) Applicant: **SUN MICROSYSTEMS, INC.**
2550 Garcia Avenue
Mountain View, CA 94043(US)

(72) Inventor: **Lockwood, James M.**
3305 Jarvis Avenue
San Jose, California 95118(US)

(74) Representative: **Wombwell, Francis**
Potts, Kerr & Co. 15, Hamilton Square
Birkenhead Merseyside L41 6BR (GB)

(54) Arbitrating multiprocessor accesses to shared resources.

(57) In a multiprocessor computer system, an access request and an access grant register is provided for storing an access request and an access grant semaphore for each shared resource. The access request and grant semaphores having a number of access request and grant bits assigned to the processors. Additionally, circuits are provided for each access request register for setting/clearing individual access request bits, and simultaneous reading of all access request bits of the stored access request semaphore. Furthermore, coordinated request and grant masks that reflect the relative access priorities of the processors are provided for the processors to use in conjunction with the current settings of the access request and grant semaphores to determine whether a shared resource is granted to a lower priority processor and whether a shared resource is being requested by a higher priority processor. As a result, multiprocessor accesses to shared resources are arbitrated in a manner having a number of advantages over the prior art.

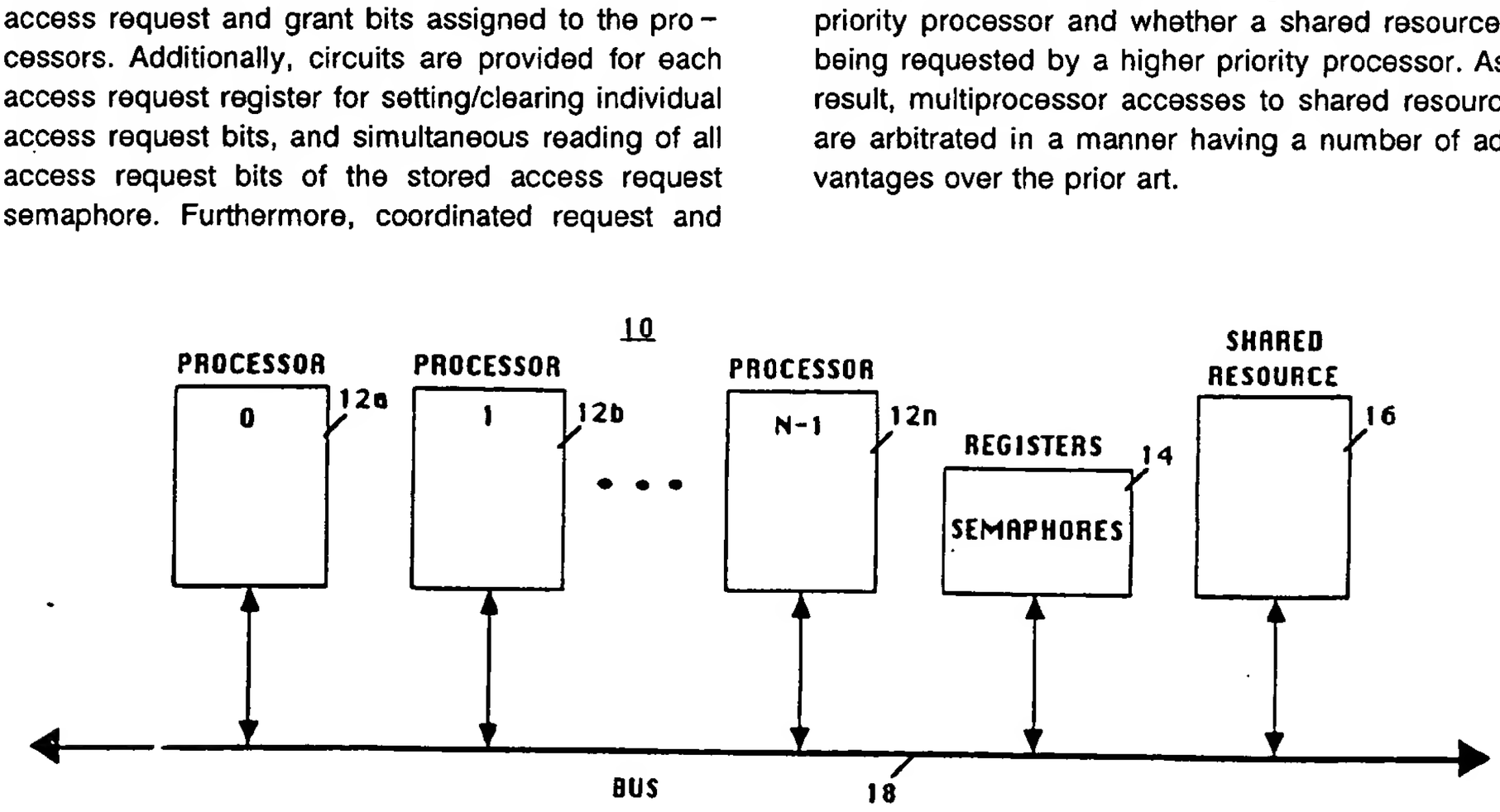

FIGURE 1

EP 0 543 560 A2

BACKGROUND OF THE INVENTION

1. Field of the Invention:

The present invention relates to the field of computer systems, in particular, multiprocessor computer systems. More specifically, the present invention relates to arbitrating multiprocessor accesses to shared resources, which has particular application to multiprocessor communication.

2. Background:

Today, many computer systems comprise multiple processors. Typically, the processors share a number of resources, for example, an area in memory for inter-processor communication. For many of these multiprocessor systems, the processors access the shared resources through a common high speed bus. Arbitration function is provided to grant momentary exclusive ownership of the bus to a particular processor. For those multiprocessor systems having private cache memory in some of the processors, coherent data management function is also provided for accessing the shared resources. The arbitration and coherent date management functions are either provided in a centralized manner by the bus or in a decentralized manner by the processors.

Additionally, some form of semaphore is employed to control access to each of the shared resources. A semaphore is a mutual exclusion lock. A processor reads the semaphore and checks if the semaphore equals a predetermined value, for example, zero, to determine if a shared resource is currently free. If the shared resource is free, the processor takes temporary ownership of the shared resource and writes another predetermined value back for the semaphore, for example, a non-zero value, to indicate such temporal taking of ownership. If the shared resource is not free, the processor repeats the reading and checking until the semaphore contains the predetermined value indicating the shared resource is free. At the end of the access, the processor reset the semaphore to indicate the shared resource is free again. The semaphore management protocol also includes mechanism or rule that the semaphore is reset only by the last processor that took temporal ownership of the shared resource. To ensure the read-modify-write manipulation of a semaphore by a processor is uninterfered by other processor trying to gain temporal ownership of the shared resource, typically the bus provides atomic bus cycles by which the processor can gain temporal ownership of the bus for a number of uninterrupted contiguous bus cycles.

Traditional semaphore management has at least four disadvantages:

1) most semaphore management tend to prioritize processor accesses on a first come first serve basis,

2) for those semaphore management that allow relative access priority settings, they generally do not provide a simple way for the contending processors to determine their own access priority level, or for the relative access priority to be changed dynamically,

3) they generally do not provide a simple way for determining which processor is the current owner of a shared resource,

4) they require all processors to be able to perform the read-modify write operations and the bus to support atomic bus cycles, furthermore, the read-modify-write operations are performed by the processors in like manner, thereby excluding the possibility of incorporating heterogeneous processors in the same computer system.

As will be disclosed, the present invention provides a method and apparatus for arbitrating multiprocessor accesses to shared resources that overcomes the disadvantages of the prior art.

SUMMARY OF THE INVENTION

It is therefore the object of the present invention to provide a method and apparatus for arbitrating multiprocessor accesses to shared resources which can be applied to multiprocessor communication.

It is another object of the present invention that accesses to shared resources are arbitrated based on relative access priority which can be altered dynamically.

It is another object of the present invention that a contending processor is able to determine its own access priority and the current owner of a shared resource.

It is another object of the present invention that the high speed bus does not have to support atomic bus cycles, and the processors do not have to perform the read-modify-write operations in like manner or support the read-modify-write operations at all, thereby enabling the incorporation of heterogeneous processors in the same computer system.

Under the preferred embodiment of the present invention, these and other objectives are achieved by providing an access request register and an access grant register for storing an access request semaphore and an access grant semaphore for each of the shared resources respectively on a multiprocessor computer system. The access request and grant registers are designed to be suffi-

ciently large enough to store access request and grant semaphores having an access request bit and an access grant bit assigned to each of the processors. In their presently preferred form, the access request and grant bits are assigned to the processors in ascending order starting with the least significant bit being assigned to the lowest priority processor.

Additionally, under the preferred embodiment of the present invention, corresponding circuits that allows setting/clearing individual access request bits and simultaneous reading of all access request bits are provided for accessing the access request semaphores stored in the access request registers.

Furthermore, a request mask and a grant mask for each of the shared resources is provided and assigned to each of the processors. The request and grant masks reflect the relative access priorities of the processors. The request and grant masks are used by the processors in conjunction with the current settings of the access request and grant semaphores to determine whether the shared resource is granted to a lower priority processor and whether the shared resource is being requested by a higher priority processor.

A processor makes its request for gaining temporal exclusive ownership of a shared resource by setting its assigned access request bit of the access request semaphore. The requesting processor then waits a predetermined amount of time. The minimum predetermined amount of time is designed to be greater than the largest amount of time required by the slowest processor to recognize the request and to set its assigned access grant bit of the access grant semaphore.

Upon waiting the predetermined amount of time, the requesting processor then repeatedly check the access grant and request semaphores to ensure there are no lower access priority grant nor higher access priority request active. Upon making such determination, the requesting processor then sets its access grant bit of the access grant semaphore and takes temporal exclusive ownership of the shared resource. After gaining temporal exclusive ownership, the owning processor performs its desired operations against the shared resource. Upon completion of these operations, the owning processor unsets its assigned access grant and request bits of the access grant and request semaphores.

As a result, the processors of a multiprocessor computer system may perform interprocessor communication via the shared resource. A contending processor may determine its relative access priority and the current owner based on the access request and grant semaphores. The relative access priorities of the processors to a shared resource may also be changed dynamically by imposing a new set of request and grant masks that reflect different relative access priorities. The need for the high speed bus and the processors to support atomic access cycles, and for the processors to support read-modify-write operations in like manner is eliminated. As a result, heterogeneous processors may be incorporated in the same multiprocessor computer system.

BRIEF DESCRIPTION OF THE DRAWINGS

The objects, features, and advantages of the present invention will be apparent from the following detailed description of the preferred embodiment of the invention with references to the drawings in which:

Figure 1 illustrates the preferred embodiment of a multiprocessor computer system that incorporates the teachings of the present invention.

Figure 2 illustrates the semaphores of the present invention used to arbitrate processor accesses to a shared resource on the multiprocessor computer system illustrated in **Figure 1**.

Figure 3 illustrates the access request masks of the present invention used to arbitrate processor accesses to a shared resource on the multiprocessor computer system illustrated in **Figure 1**.

Figure 4 illustrates the access grant masks of the present invention used to arbitrate processor accesses to a shared resource on the multiprocessor computer system illustrated in **Figure 1**.

Figures 5a – 5b illustrate two embodiments of the circuit of the present invention for setting/clearing individual bits and simultaneous readings of all request bits of the request semaphores stored in an access request register on the multiprocessor computer system illustrated in **Figure 1**.

Figure 6 illustrates the method of the present invention for arbitrating processor accesses to a shared resource on the multiprocessor computer system illustrated in **Figure 1**.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A method and apparatus for arbitrating processor accesses to shared resources on a multiprocessor computer system is disclosed, which has particular application to multiprocessor communication. In the following description for purposes of explanation, specific numbers, materials and configurations are set forth in order to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art

that the present invention may be practiced without the specific details. In other instances, well known systems are shown in diagrammatical or block diagram form in order not to obscure the present invention unnecessarily.

Referring now to **Figure 1**, a block diagram illustrating the preferred embodiment of a muitiprocessor computer system that incorporates the teachings of the present invention is shown. Shown are a plurality of processors 12a - 12n. Each of the processors 12a - 12n comprise an request mask register and a grant mask register for storing a request mask and a grant mask respectively, which will be described in further detail below. The processors 12a - 12n are intended to represent a broad category of well known processors. In fact, under the preferred embodiment, processors 12a - 12n may be a mix of different well known processors that do not support read-modify-write operations in like manner or do not support read-modify-write operations at all. Particular examples of these processors 12a - 12n include SPARC™ processors manufactured by Sun Microsystems, Inc., Mountain View, Ca., and i486™ processors manufactured by Intel, Inc., Santa Clara, Ca. (SPARC™ is a registered trademark of Sun Microsystems, Inc, and i486™ is a registered trademark of Intel, Inc.).

The processors 12a - 12n are coupled to an active high speed bus 18. The active high speed bus 18 provides centralized arbitration function for granting temporal exclusive ownership of the bus 18. Additionally, the active high speed bus 18 provides coherent data management functions for the processors 12a - 12n if some of the processors 12a - 12n comprise private cache memory. However, under the presently preferred embodiment, the active high speed bus 18 does not have to provide atomic bus cycles whereby a processor may own the bus 18 for a number of uninterrupted contiguous bus cycles. The active high speed bus 18 is also intended to represent a broad category of well known active high speed buses. In fact, active high speed bus 18 may comprise multiples of these well known buses coupled together by bus adaptors. Particular examples of the well known high speed buses include the MBus of Sun Microsystems, Inc., and the SBus. Although an active high speed bus is preferred, it will be appreciated that high speed bus 18 may also be an inactive bus, with the bus arbitration and coherent data management functions distributed to the various processors 12a - 12n.

Also shown is a shared resource 16 and its corresponding semaphore registers 14 coupled to the active high speed bus 18. The shared resource 16 is intended to represent a broad category of resources on a multiprocessor computer system. For example, the shared resource 16 may be an area of main memory used as a shared message buffer for interprocessor communication between the processors 12a - 12n. The corresponding semaphore registers 14 comprise an access request register (not shown), an access grant register (not shown) and corresponding circuits (not shown) for writing into and reading from the registers. The semaphore registers 14 will be described in further detail below. Although only one shared resource 16 is illustrated in **Figure 1**, it will be appreciated that the multiprocessor computer system 10 may comprise a number of shared as well as unshared resources, provided each of the additional shared resources also has its own corresponding semaphore registers. Furthermore, although the semaphore registers 14 are preferred, it will also be appreciated that memory locations in memory may be used instead, provided the active high speed bus 18 supports atomic bus cycles for performing read-modify-write operations into the memory locations, and the processors 12a - 12n support read-modify-write operations in like manner.

Referring now to **Figure 2**, a block diagram illustrating the access request register, the access grant register and the corresponding access request and grant semaphores stored in these registers is shown. Shown is an access request register 20 storing an access request semaphore having a plurality of access request bits 24a - 24n. Each of the access request bits 24a - 24n is assigned to a processor. In its presently preferred form, the access request bits 24a - 24n are assigned to the processors in ascending order starting with the least significant bit assigned to the processor with the lowest priority. It will be appreciated that the access request bits 24a - 24n may be assigned in a variety of other manners.

Also shown is a grant access register 22 storing an access grant semaphore having a plurality of access grant bits 26a - 26n. Each of the corresponding access grant bits 26a - 26n is assigned to the same processor. Similarly, in its presently preferred form, the access grant bits 26a - 26n are also assigned to the processors in ascending order starting with the least significant bit assigned to the processor with the lowest priority. It will also be appreciated that the access grant bits 26a - 26n may be assigned in a variety of other manners, provided it is consistent with the way the access request bits 24a - 24n are granted.

Each of the access request bits 24a - 24n is set individually by its assigned processor when its assigned processor request access to the shared resource, which will be described in further detail below. Multiple access request bits 24a - 24n may be set by a number of requesting processors

at one time. Each of the access grant bits 26a – 26n is set by its assigned processor when its assigned processor after requesting accessing to the shared resource determines that the shared resource is neither granted to a lower priority processor nor being requested by a higher priority processor, which will also described in further detail below. It will be appreciated that only one access grant bit 26a, 26b, ...or 26n, will be set by one processor at one time. As a result, the current owner of the shared resource can be determined based on the access grant semaphore easily.

Each of the access grant bits 26a – 26n is unset by its assigned processor after its assigned processor completed access to the shared resource. Each of the access request bits 24a – 24n is unset by its assigned processor after its assigned processor unset the corresponding access grant bit, 26a, 26b, ...or 26n.

A requesting processor determines whether the shared resource is granted to a lower priority processor and whether the shared resource is being requested by a higher priority processor using the access request and grant semaphores in conjunction with an access request mask and an access grant mask, which will be described in further detail below. It will be appreciated that under the present invention, a requesting processor does not have to concern itself with whether the shared resource is granted to a higher priority processor nor whether the shared resource is being requested by a lower priority processor.

Referring now to **Figure 3**, a block diagram illustrating the access request masks of the present invention is shown. Shown are a plurality of access request masks 28a – 28n comprising a plurality request mask bits 30a – 30n. Each of the request masks 28a – 28n comprises the same number of bits as the access request semaphore. For each request mask, 28a, 28b, ..., or 28n, the request mask bits 30a – 30n that are in higher bit positions than the request bit of the access request semaphore assigned to the processor are set, and all other request mask bits are unset. For examples, for processor 0, request mask bits 1 through n are set and request mask bit 0 is unset; for processor 1, request mask bits 1 through n are set and request mask bits 0 through 1 are unset; and for processor n, all request mask bits 0 through n are unset.

The request masks 28a – 28n are stored in the private request mask registers of the processors. A requesting processor determines whether the shared resource is being requested by a higher priority processor by performing a logical AND against the current bit settings of the access request semaphore and its request mask. The request masks 28a – 28n may be changed dynamically, thereby altering the relative access priorities of the processors dynamically, provided the grant masks which are described below are also changed at the same time.

Referring now to **Figure 4**, a block diagram illustrating the access grant masks of the present invention is shown. Shown are a plurality of access grant masks 32a – 32n comprising a plurality grant mask bits 34a – 34n. Each of the grant masks 32a – 32n comprises the same number of bits as the access grant semaphore. For each grant mask, 32a, 32b, ..., or 32n, the grant mask bits 32a – 32n that are in lower bit positions than the grant bit of the access grant semaphore assigned to the processor are set, and all other grant mask bits are unset. For examples, for processor 0, all grant mask bits 0 through n are unset; for processor 1, grant mask bits 0 is set and grant mask bits 1 through n are unset; and for processor n, grant mask bits 0 through n – 1 are set and grant mask bits 0 is unset.

Similarly, the grant masks 32a – 32n are stored in the private grant mask registers of the processors. A requesting processor determines whether the shared resource is granted to a lower priority processor by performing a logical AND against the current bit settings of the access grant semaphore and its grant mask. As described earlier, the grant masks 28a – 28n may be changed dynamically, thereby altering the relative access priorities of the processors dynamically, provided the request masks are also changed at the same time.

Referring now to **Figures 5a** and **5b**, two block diagrams illustrating two embodiments of the circuit for setting/clearing individual access request bits, and simultaneous reading of all access request bits of the access request semaphores stored in the access request registers are shown. The first embodiment illustrated in **Figure 5a** is designed for multiprocessor computer systems that do not have processor identifiers presented on the high speed bus, whereas, the second embodiment illustrated in **Figure 5b** is designed for multiprocessor computer systems that presents the processor identifiers on the high speed bus. As described earlier, only one access grant bit of the access grant semaphore is set at one time, thus no special circuit for writing and reading the access grant semaphore into and from the access grant register is required. The access grant register may be implemented with any well known manner in the art.

Shown in **Figure 5a** are identical circuit segments 42a – 42n. Each of the identical circuit segments 42a – 42n is used for setting, clearing, and reading the access request bits of the access request semaphores assigned to a processor. Each of the identical circuit segments 42a – 42n com-

prises a first AND gate 48a, 48b, ..., or 48n, a second AND gate 50a, 50b ..., or 50n, and a tri-state buffer 44a, 44b ..., or 44n coupled to a latch 46a, 46b ..., 46n, of the access request register. The AND gates 48a - 48n, 50a - 50n, the tri-state buffers 44a - 44n, and the latches 46a - 46n are well known in the art.

The first AND gate 48a, 48b ..., or 48n, is for setting the assigned access request bit of the access request semaphore. The first AND gate 48a, 48b, ..., or 48n, takes a data line, a first write enable signal, and a strobe signal from the high speed bus as inputs, and outputs a signal into the latch 46a, 46b, ..., 46n. In their presently preferred form, if output value is a "1", the access request bit is set; if the output value is a "0", no action is taken.

Similarly, the second AND gate 50a, 50b, ..., or 50n, is for clearing the assigned access request bit of the access request semaphore. The second AND gate 50a, 50b ..., or 50n, takes a data line, a second write enable signal, and a strobe signal from the high speed bus as inputs, and outputs a signal into the latch 46a, 46b, ..., 46n. In their presently preferred form, if output value is a "1", the access request bit is unset; if the output value is a "0", no action is taken.

The latch 46a, 46b, ..., or 46n takes either the output of the first AND gate 48a, 48b, ..., or 48n, or the output of the second AND gate 50a, 50b, ..., or 50n as input, and outputs a signal into the tri-state buffer 44a,44b, ..., or 44n. The tri-state buffer 44a,44b, ..., or 44n, takes the output of the latch 46a, 46b ..., or 46n, and a read enable signal from the high speed bus as inputs and output a signal onto the data line of the high speed bus.

Shown in **Figure 5b** is an AND gate 66 and a tri-state buffer 62 coupled to the access request register which is implemented by an addressable latch 64. The AND gate 66, the tri-state buffer 62 and the addressable latch 64 are also well known in the art. The AND gate 66 takes a write enable signal and a strobe signal from the high speed bus as input and outputs a signal to the addressable latch. The addressable latch 64 takes the output of the AND gate 66, a data line from the high speed bus, and the processor identifier from the high speed bus as inputs, and outputs a plurality of signals to the tri-state buffer 62. The processor identifier is used to select the access request/grant bit for replacement with the input from the data line. The tri-state buffer 62 takes the output signals of the addressable latch 64 and a read enable signal from the high speed bus as input and outputs a plurality of signals on the high speed bus.

It will be appreciated by providing one of the circuits illustrated in **Figures 5a** and **5b**, the access request semaphore can be manipulated without requiring support of atomic bus cycles by the high speed bus, whereby the requesting processor can have temporal exclusive ownership of the high speed bus for a number of uninterrupted clock cycles. Furthermore, the processors are not required to support read-modify-write operations in like manner or even support read-modify-write operations at all. As a result, heterogeneous processors may be incorporated in the same multiprocessor computer system.

Referring now to **Figure 6**, a block diagram illustrating the method of the present invention for arbitrating processor accesses to shared resources is shown. As shown in the figure, a requesting processor request access to the shared resource by setting its assigned request bit of the access request semaphore, block 72. The requesting processor then waits a pre-determined amount of time, block 74. The minimum pre-determined amount of wait time is greater than the largest amount of time required by the slowest processor to recognize the request, and to set its assigned access grant bit of the access grant semaphore. As described earlier, the processor recognizes the request by retrieving the current bit settings of the access request semaphore from the access request register and performing a logical AND against the retrieved bit settings and the processor's request mask.

Upon waiting at least the minimum pre-determined amount of time, the requesting processor determines if the shared resource is granted to a lower priority processor, block 76. As described earlier, the requesting processor makes the determination by retrieving the current bit settings of the access grant semaphore from the access request register and performing a logical AND against the retrieved bit settings and the processor's grant mask. If the processor is currently granted to a lower priority processor, the requesting processor repeats the determination until the shared resource is no longer granted to a lower priority processor.

Upon determining that the shared resource is not granted to a lower priority processor, the requesting processor determines if the shared resource is being requested by a higher priority processor, block 78. Similarly, if the shared resource is being requested by a higher priority processor, the requesting processor repeats the determination until the shared resource is not being requested by a higher priority processor.

Upon determining that the shared resource is not being requested by a higher priority processor, the requesting processor then sets the assigned access grant bit of the access grant semaphore, block 80, and takes temporal exclusive ownership of the shared resource. The granted processor performs its desired operations against the shared

resource, block 82, for example, writing a message if the shared resource is a shared message buffer.

Upon completion of the operations against the shared resource, the granted processor unsets the assigned access grant bit of the access grant semaphore, block 84. After unsetting the assigned access grant bit, the ungranted processor unsets the assigned access request bit of the access request semaphore, block 86.

While the present invention has been described in terms of a presently preferred embodiment, those skilled in the art will recognize that the invention is not limited to the embodiment described. The method and apparatus of the present invention can be practiced with modification and alteration within the spirit and scope of the appended claims. The description is thus to be regarded as illustrative instead of restrictive on the present invention.

Claims

1. In a computer system comprising a plurality of processors coupled to a plurality of shared resources, a method for arbitrating accesses to a shared resource by said processors, said method comprising the steps of:

   a) setting an assigned access request bit of an access request semaphore by a requesting processor, each of said processors being assigned an access request bit of said access request semaphore;

   b) waiting a predetermined amount of time;

   c) retrieving current bit settings of an access grant semaphore and determining if said shared resource is currently granted to a lower priority processor by said requesting processor, each of said processors being assigned an access grant bit of said access grant semaphore;

   d) retrieving current bit settings of said access request semaphore and determining if said shared resource is currently requested by a higher priority processor by said requesting processor when said shared resource is not granted to a lower priority processor;

   e) setting said assigned grant bit in said access grant semaphore by said requesting processor when said shared resource is not requested by a higher priority processor;

   f) accessing said shared resource by said granted processor; and

   g) unsetting said assigned grant bit of said access grant semaphore by said accessing processor upon completion of accessing said shared resource; and

   h) unsetting said assigned request bit of said access request semaphore by said ungranted processor upon unsetting of said bit position of said access grant semaphore.

2. The method as set forth in claim 1, wherein said access request and grant bits of said access request and grant semaphores are assigned to said processors in ascending order starting with the least significant bit being assigned to the lowest priority processor.

3. The method as set forth in claim 2, wherein said requesting processor determines if said shared resource is currently granted to a lower priority processor by performing a logical AND on said current bit settings of said access grant semaphore and an assigned access grant mask, said assigned access grant mask comprising a plurality of grant mask bits corresponding to said access grant semaphore, said grant mask bits being set in all corresponding bits lower than said assigned access grant bit of said requesting processor and unset in all other corresponding access grant bits.

4. The method as set forth in claim 3, wherein, said requesting processor determines if said shared resource is currently being requested by a higher priority processor by performing a logical AND on said current bit settings of said access request semaphore and an assigned access request mask, said assigned access request mask comprising a plurality of request mask bits corresponding to said access request register, said request mask bits being set in all corresponding access request bits higher than said assigned access request bit of said requesting processor and unset in all other corresponding access request bits.

5. The method as set forth in claim 1, wherein,

   said access request and grant semaphores are stored in a first and a second memory location of memory, said computer system further comprising said memory; and

   said processors, said shared resource and said memory are coupled to a high speed bus, said high speed bus providing atomic bus cycles for accessing said first and second memory locations.

6. The method as set forth in claim 1, wherein,

   said access request and grant semaphores are stored in an access request register and an access grant register respectively, said requesting processor setting and unsetting said

bit positions, and retrieving said current bit settings, of said access request and grant semaphores stored in said access request and grant registers, using circuits that allow setting and clearing of individual access request and grant bits, and simultaneous reading of all access request and grant bits of said access request and grant semaphores stored in said access request and grant registers, said computer system further comprising said circuits; and

said processors, said shared resource, said access request and grant registers, and said circuits are coupled to a high speed bus.

7. The method as set forth in claim 6, wherein,

said high speed bus does not provide processor identifiers for said processors;

said access request and grant registers are implemented with a plurality of latches; and

each of said circuits comprises a plurality of tri-state buffers, a first and second plurality of AND gates.

8. The method as set forth in claim 6, wherein,

said high speed bus provides processor identifiers for said processors;

said access request and grant registers are implemented with addressable latches; and

each of said circuits comprises a tri-state buffer, and an AND gate.

9. The method as set forth in claim 1, wherein, said predetermined amount of time is greater than the largest amount of time required by the slowest of said processors to perform said steps c), d) and e).

10. The method as set forth in claim 1, wherein, said shared resource is an area in memory used for inter-processor communication between said processors.

11. In a computer system comprising a plurality of processors coupled to a plurality of shared resources, an apparatus for arbitrating accesses to a shared resource by said processors, said apparatus comprising:

a) first storage means coupled to said processors for storing an access request semaphore comprising a plurality of access request bits, each of said access request bits being assigned to one of said processors, each of said assigned access request bits being set by its processor when its processor requests access to said shared resource; and

b) second storage means coupled to said processors for storing an access grant semaphore comprising a plurality of access grant bits, each of said access grant bits being assigned to one of said processors, each of said assigned access grant bits being set by its processor when its processor after requesting access to said shared resource determines that said shared resource is not granted to a lower priority processor nor being requested by a higher priority processor, said determinations being made by said requesting processor based on current bit settings of said access grant and request semaphores, said determinations being made after said requesting processor has waited a predetermined amount of time after requesting access to said shared resource;

each of said set access grant bits being unset by its processor when its processor complete accessing said shared resource, each of said set access request bits being unset by its processor after its processor unset its set access grant bit.

12. The apparatus as set forth in claim 11, wherein said access request and grant bits of said access request and grant semaphores are assigned to said processors in ascending order starting with the least significant bit being assigned to the lowest priority processor.

13. The apparatus as set forth in claim 12, wherein said requesting processor determines if said shared resource is currently granted to a lower priority processor by performing a logical AND on said current bit settings of said access grant semaphore and an assigned access grant mask, said assigned access grant mask comprising a plurality of grant mask bits corresponding to said access grant semaphore, said grant mask bits being set in all corresponding grant mask bits lower than said assigned access grant bit of said requesting processor and unset in all other corresponding grant mask bit positions.

14. The apparatus as set forth in claim 13, wherein, said requesting processor determines if said shared resource is currently being requested by a higher priority processor by performing a logical AND on said current bit settings of said access request semaphore and an assigned access request mask, said assigned access request mask comprising a plurality of request mask bits corresponding to said access request register, said request

mask bits being set in all corresponding request mask bits higher than said assigned access request bit of said requesting processor and unset in all other corresponding request mask bits.

15. The apparatus as set forth in claim 11, wherein,

said first and second storage means are the same memory means coupled to said processors comprising a first and second memory locations for storing said access request and grant semaphores, said computer system further comprising said memory; and

said processors, said shared resource and said memory are coupled to a high speed bus, said high speed bus providing atomic bus cycles for accessing said first and second memory locations.

16. The apparatus as set forth in claim 11, wherein,

said first and second storage means are the same register means coupled to said processors comprising an access request register and an access grant register for storing said access request and grant semaphores respectively;

said register means further comprises circuits coupled to said processors and said access request and grant registers for said requesting processor to set and unset said access request and grant bits, and to retrieve said current bit settings of said access request and grant semaphores stored in said access request and grant registers, said circuits allowing setting and clearing of individual access request and grant bits, and simultaneous reading of all access request and grant bits of said access request and grant semaphores stored in said access request and grant registers; and

said processors, said shared resource, said access request and grant registers, and said circuits are coupled to a high speed bus.

17. The apparatus as set forth in claim 16, wherein,

said high speed bus does not provide processor identifiers for said processors;

said access request and grant registers are implemented with a plurality of latches; and

each of said circuits comprises a plurality of tri-state buffers coupled to said high speed bus and said latches, a first and second plurality of AND gates coupled to said high speed bus and said latches.

18. The apparatus as set forth in claim 16, wherein,

said high speed bus provides processor identifiers for said processors;

said access request and grant registers are implemented with addressable latches; and

each of said circuits comprises a tri-state buffer coupled to said high speed bus and said addressable latch, and an AND gate coupled to said addressable latch and said high speed bus.

19. The apparatus as set forth in claim 11, wherein, said predetermined amount of time is greater than the largest amount of time required by the slowest of said processors to perform said determinations and to set its assigned access grant bit of said access grant semaphore.

20. The apparatus as set forth in claim 11, wherein, said shared resource is an area in memory coupled to said processors used for inter-processor communication between said processors, said computer system further comprising said memory.

21. A multiprocessor computer system comprising:

a high speed bus;

a plurality of processors coupled to said high speed bus;

a shared resource coupled to said high speed bus; and

an apparatus for arbitrating processor accesses to said shared resource comprising first and second storage means for storing an access request and an access grant semaphore, said access request and grant semaphores comprising a plurality of access request and access grant bits respectively, each of said processors being assigned one of said access request bits and a corresponding access grant bits, each of said assigned access request bits being set by its processor when its processor requests access to said shared resource, each of said assigned access grant bit being set by its processor when its processor after requesting said access to said shared resource determines that said shared resource is not granted to a lower priority processor nor being requested by a higher priority processor;

said determinations being made by said requesting processor based on current bit settings of said access grant and request semaphores, said determinations being made after said requesting processor has waited a predetermined amount of time after requesting access to said shared resource;

each of said set access grant bits being unset by its processor when its processor has completed access to said shared resource, each of said set access request bits being unset by its processor after its processor unset its set grant bit.

22. In a computer system comprising a plurality of processors, a method for performing multiprocessor communications among said processors, said method comprising the steps of:

a) setting an assigned access request bit of an access request semaphore for a shared message buffer by a requesting processor, each of said processors being assigned an access request bit of said access request semaphore;

b) waiting a predetermined amount of time;

c) retrieving current bit settings of an access grant semaphore for said shared message buffer and determining if said shared message buffer is currently granted to a lower priority processor by said requesting processor, each of said processors being assigned an access grant bit of said access grant semaphore;

d) retrieving current bit settings of said access request semaphore and determining if said shared message buffer is currently requested by a higher priority processor by said requesting processor when said shared message buffer is not granted to a lower priority processor;

e) setting said assigned access grant bit in said access grant semaphore by said requesting processor when said shared message buffer is not requested by a higher priority processor;

f) writing a message into said shared message buffer by said granted processor; and

g) unsetting said assigned access grant bit of said access grant semaphore by said accessing processor upon completion of writing a message into said shared message buffer; and

h) unsetting said assigned access request bit of said access request semaphore by said ungranted processor upon unsetting of said access grant bit of said access grant semaphore.

23. In a computer system comprising a plurality of processors, an apparatus for performing multiprocessor communication among said processors, said apparatus comprising:

a) a shared message buffer coupled to said processors;

b) first storage means coupled to said processors for storing an access request semaphore for said shared message buffer, said access request semaphore comprising a plurality of access request bits, each of said access request bits being assigned to one of said processors, each of said assigned access request bits being set by its processor when its processor requests access to said shared message buffer; and

b) second storage means coupled to said processors for storing an access grant semaphore for said shared message buffer, said access grant semaphore comprising a plurality of access grant bits, each of said access grant bits being assigned to one of said processors, each of said assigned access grant bits being set by its processor when its processor after requesting access to said shared message buffer determines that said shared message buffer is not granted to a lower priority processor nor being requested by a higher priority processor, said determinations being made by said requesting processor based on current bit settings of said access grant and request semaphores, said determinations being made after said requesting processor has waited a predetermined amount of time after requesting access to said shared message buffer;

each of said set access grant bits being unset by its processor when its processor complete writing a message into said shared message buffer, each of said set access request bits being unset by its processor after its processor unset its set grant bit.

24. A multiprocessor computer system comprising:

a high speed bus;

a plurality of processors coupled to said high speed bus;

a shared message buffer coupled to said high speed bus; and

an apparatus for performing multiprocessor communication using said shared message buffer, said apparatus comprising first and second storage means for storing an access request and an access grant semaphore for said shared message buffer, said access request and grant semaphores comprising a plurality of access request and access grant bits respectively, each of said processors being assigned one of said access request bits and a corresponding access grant bits, each of said assigned access request bits being set by its processor when its processor requests access to said shared message buffer, each of

said assigned access grant bit being set by its processor when its processor after requesting said access to said shared message buffer determines that said shared message buffer is not granted to a lower priority processor nor being requested by a higher priority processor;

said determinations being made by said requesting processor based on current bit settings of said access grant and request semaphores, said determinations being made after said requesting processor has waited a predetermined amount of time after requesting access to said shared message buffer;

each of said set access grant bits being unset by its processor when its processor complete accessing said shared message buffer, each of said set access request bits being unset by its processor after its processor unset its set grant bit.

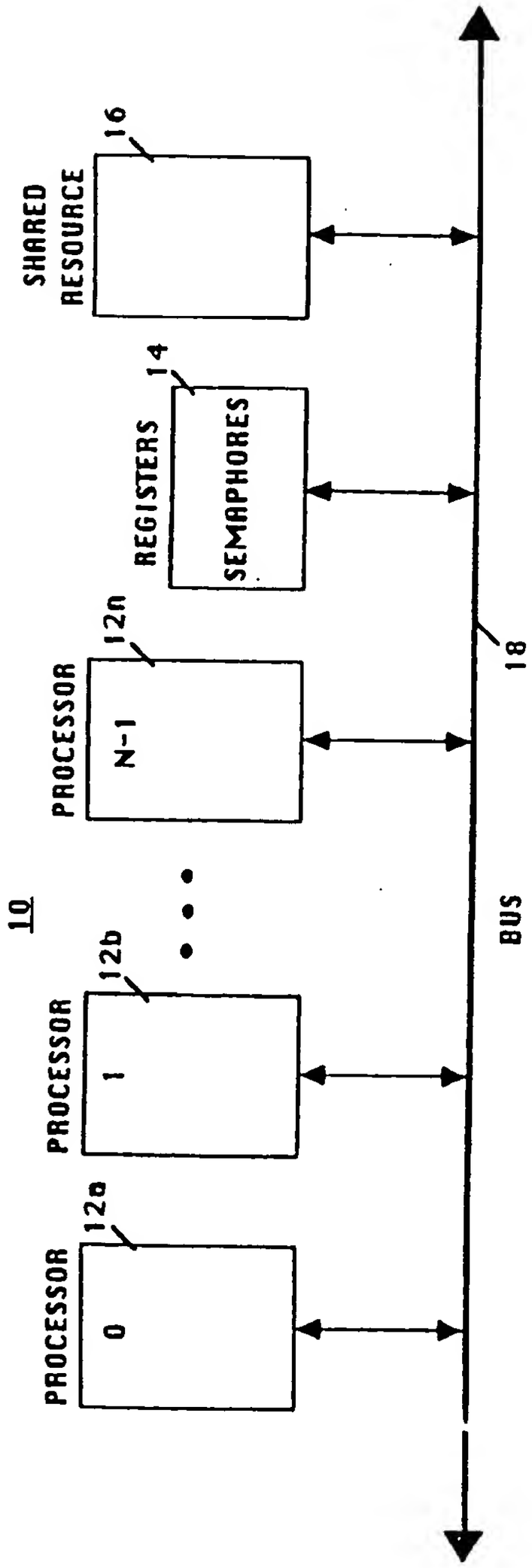
10
PROCESSOR 0
12a
PROCESSOR 1
12b
PROCESSOR N-1
12n
REGISTERS SEMAPHORES
14
SHARED RESOURCE
16
BUS
18

FIGURE 1

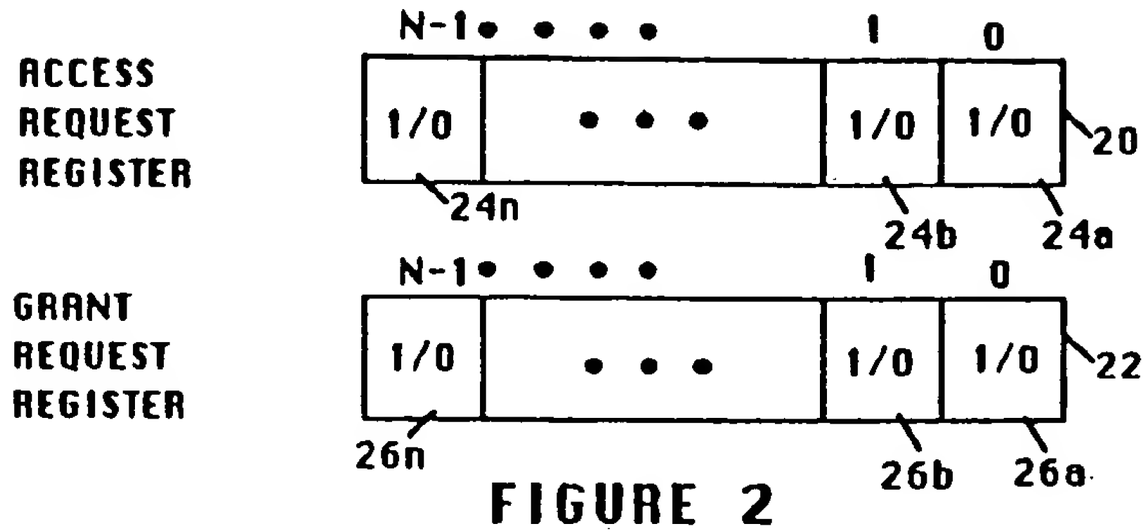

FIGURE 2

1/0

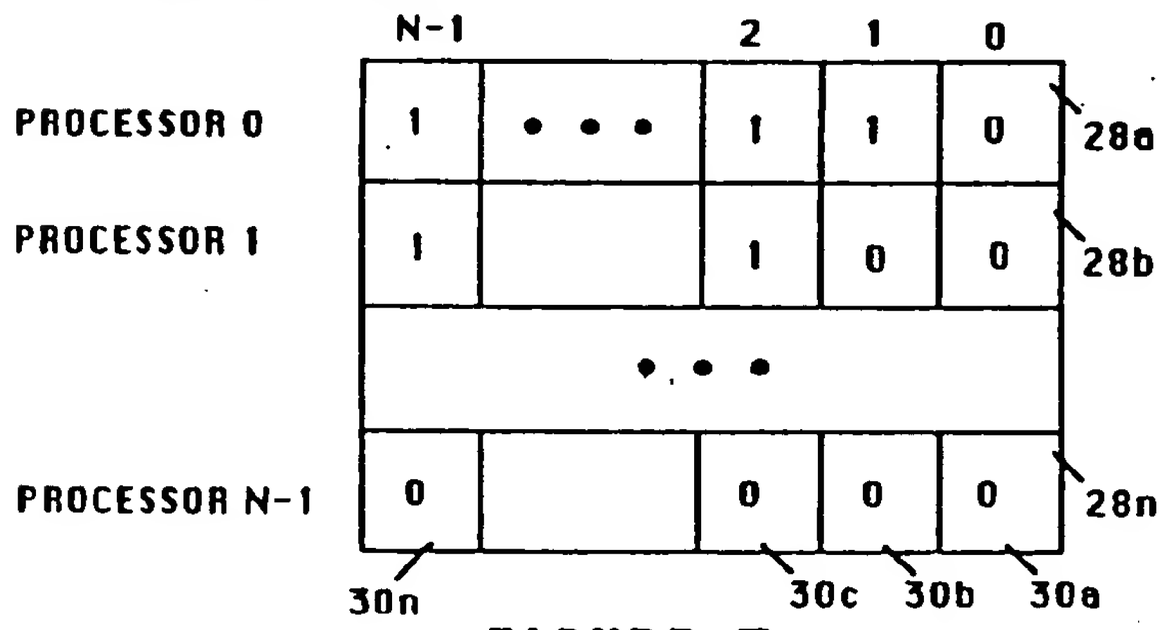

FIGURE 3

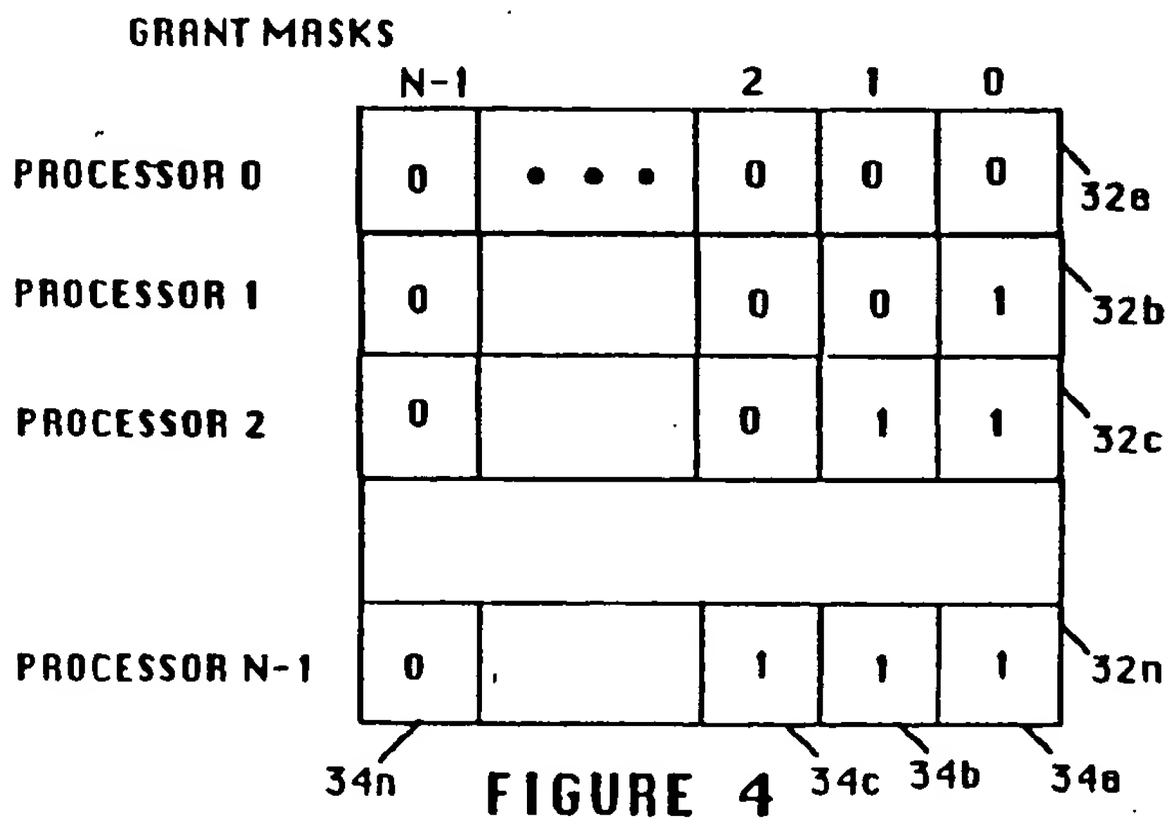

FIGURE 4

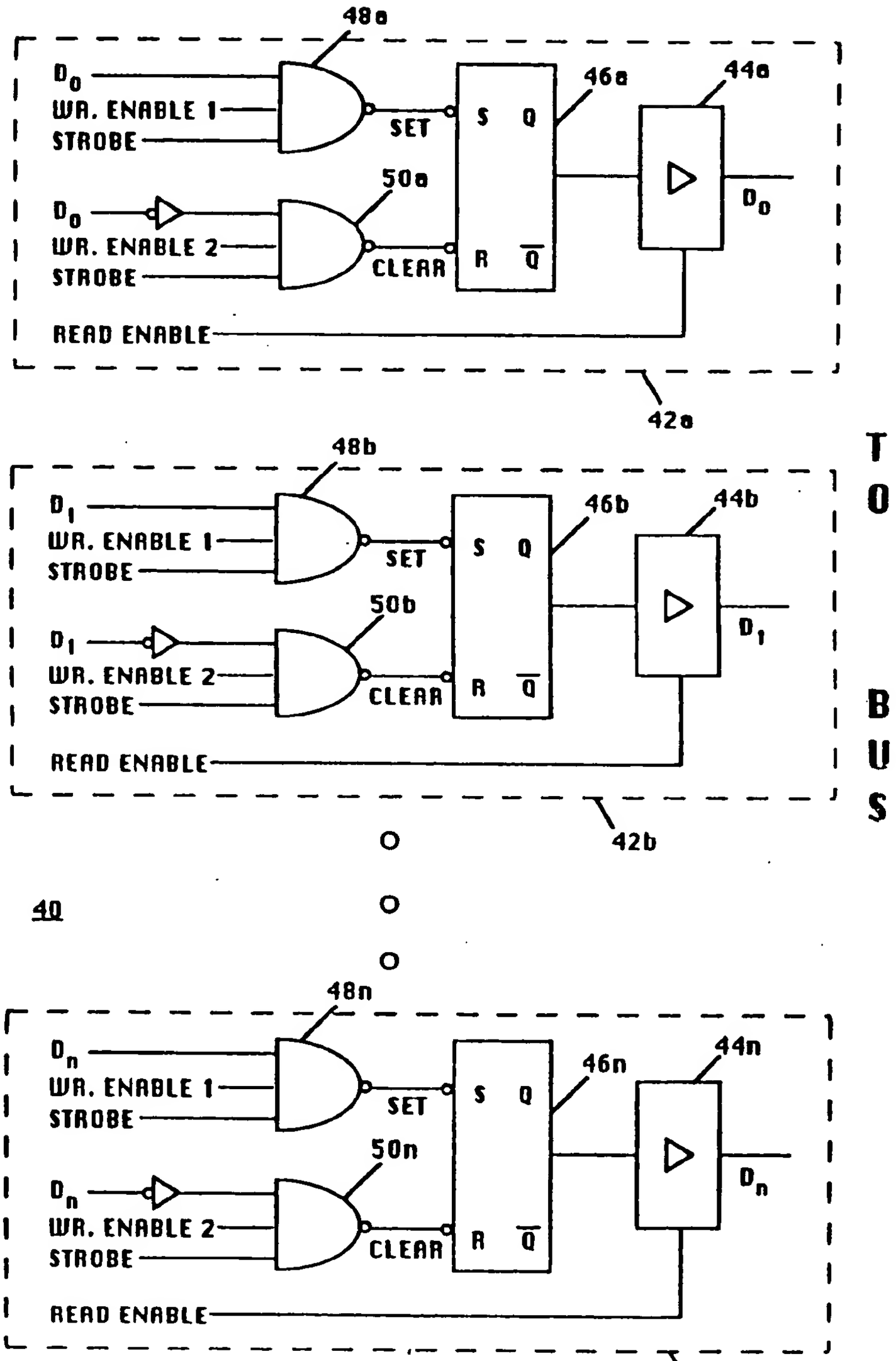
48a
D_0
WR. ENABLE 1
STROBE
SET
S Q
46a
44a
50a
D_0
WR. ENABLE 2
STROBE
CLEAR
R $\overline{Q}$
D_0
READ ENABLE
42a
48b
D_1
WR. ENABLE 1
STROBE
SET
S Q
46b
44b
50b
D_1
WR. ENABLE 2
STROBE
CLEAR
R $\overline{Q}$
D_1
READ ENABLE
42b
TO BUS
40
48n
D_n
WR. ENABLE 1
STROBE
SET
S Q
46n
44n
50n
D_n
WR. ENABLE 2
STROBE
CLEAR
R $\overline{Q}$
D_n
READ ENABLE
42n

FIGURE 5a

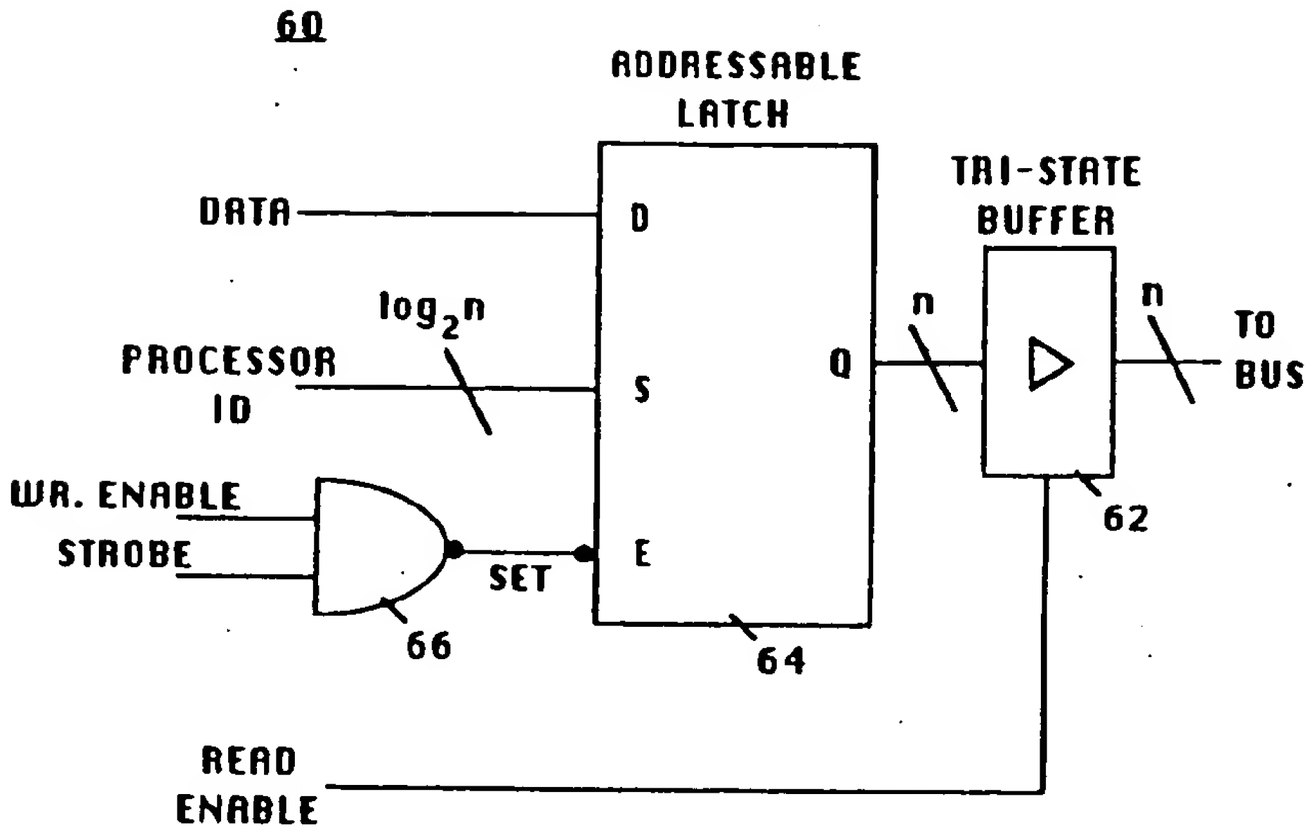
60
ADDRESSABLE LATCH
TRI-STATE BUFFER
DATA
D
PROCESSOR ID
log2n
S
Q
n
n
TO BUS
WR. ENABLE
STROBE
SET
E
66
64
62
READ ENABLE

FIGURE 5b

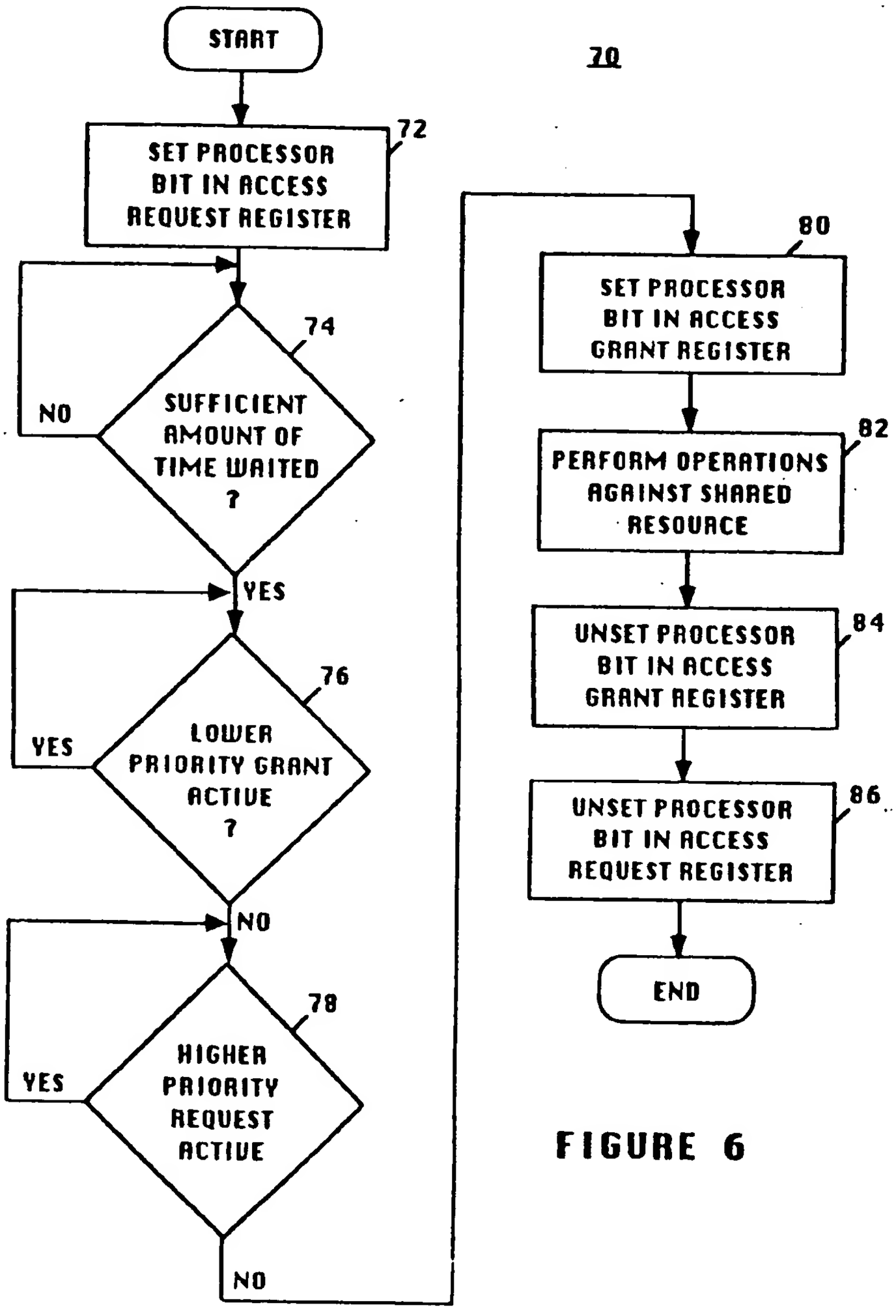
START
70
72
SET PROCESSOR BIT IN ACCESS REQUEST REGISTER
74
SUFFICIENT AMOUNT OF TIME WAITED ?
NO
YES
76
LOWER PRIORITY GRANT ACTIVE ?
YES
NO
78
HIGHER PRIORITY REQUEST ACTIVE
YES
NO
80
SET PROCESSOR BIT IN ACCESS GRANT REGISTER
82
PERFORM OPERATIONS AGAINST SHARED RESOURCE
84
UNSET PROCESSOR BIT IN ACCESS GRANT REGISTER
86
UNSET PROCESSOR BIT IN ACCESS REQUEST REGISTER
END

FIGURE 6